Improving Search Using a Fault-Tolerant Overlay in Unstructured P2P Systems*

William Acosta Surendar Chandra
University of Notre Dame
$\{wacosta, surendar\}@cse.nd.edu$

Abstract

Gnutella overlays have evolved to use a two-tier topology. However, we observed that the new topology had only achieved modest improvements in search success rates. Also, the new two-tier topology had not reduced the message routing overhead and bandwidth consumption. In this work, we used local information at each node to construct an overlay, Makalu*, that improved search performance and reduced bandwidth consumption. The overlay maximized the expansion from each node's neighborhood while minimizing the latency to its neighbors. We show that for a 100,000 node system, wild card searches using flooding successfully resolved most queries within four hops for object replications ratios as lows as 0.05% (50 randomly distributed copies) with less than 3% duplicate messages. Using attenuated bloom filters to route messages for exact identifier searches, we show that* Makalu *resolved most queries with less than ten messages for networks as large as 100,000 nodes. The performance of this search is comparable to that of structured P2P systems. Finally, using data from traffic traces of Gnutella in 2003 and 2006, we demonstrated search success rates that were up to five times more successful and required 75% less bandwidth on a Makalu overlay than on a modern Gnutella overlay.*

Keywords: P2P and Ubiquitous Computing

1. Introduction

The popularity of Gnutella [24] coupled with poor internet connections of many users [27] exposed the fundamental performance issues of the Gnutella protocol. The original protocol treated all peers equally. However, many peers with poor network connectivity could not effectively handle many connections to neighbors and could not support the bandwidth demands of routing and forwarding queries. As a result, search performance suffered; many queries experienced long queueing times or were not resolved at all. To address these problems, a new protocol (v0.6) [15] was created. The new protocol treated poorly connected nodes (leaves) differently than well connected nodes (ultra-peers). The purpose of the new architecture was to shift the burden of handling message routing away from leaf peers to more well connected ultra-peers. To this end, the majority of messages handled in Gnutella are the responsibility of the ultra-peers. Although these changes successfully addressed the issue of bandwidth utilization at the leaves, they did not address the problem of poor search performance in Gnutella. We previously studied the long-term evolution of Gnutella traffic as the network shifted from the v0.4 to v0.6 protocol [1]. Between 2003 and 2006, query success rates experienced by intermediate nodes only increased from 3.5% to 6.9%. Over 90% of all the bytes received by these intermediate nodes are wasted. Additionally, the number of messages sent per query has increased an order of magnitude since 2003.

In this paper, we developed an overlay that improved search performance in unstructured P2P networks. Our fundamental assertion is that search performance is directly affected by the connectivity properties of the underlying overlay. In addition to improving search performance, good connectivity results in an overlay that is fault-tolerant. Our algorithm, *Makalu*, relies only on local information for each node and maximizes the expansion from each node's neighborhood while minimizing the latency to its neighbors. We show that our overlay exhibited expansion properties similar to expander graphs[3, 12], which are graphs known to have good connectivity. We exploited the good connectivity of *Makalu* to solve the problem of efficient wild-card and attribute searches using flooding. In particular, we show that *Makalu* resolved most queries within four hops in a 100,000 node network while minimizing the number of duplicate messages that were generated. In addition to flooding searches, we also exploited the expander graphs properties of *Makalu* to support efficient identifier searches in unstructured P2P systems. Having access to a large set of nodes within a few hops increased the amount of information about the system available to each node. We leveraged this to implement a routing algorithm using attenuated

*This work was supported in part by the U.S. National Science Foundation (CNS-0447671).

Bloom filters [25] to handle identifier search. We show that this mechanism successfully resolved all queries with fewer than 10 messages on a 10,000 node network and fewer than 20 messages on a 100,000 node network even when replication ratios for objects are as low as 0.1%. In applications such as the Gnutella file-sharing network [14, 15] that operate with query success rates of less than 10%, we show experimentally that our overlay *Makalu* achieved higher query success rates with lower bandwidth utilization and a lower number of connections per node. Using query request rates obtained from [1], we validated our design by showing that a *Makalu* overlay resolved five times as many queries and use 75% less bandwidth than Gnutella. Further, we show that *Makalu* achieved this with fewer than 20% of neighbors required by a typical Gnutella ultra-peer.

The rest of this paper is organized as follows. Section 2 describes the design of the *Makalu* overlay algorithm and presents an analysis of the structural and connectivity properties of *Makalu* overlay topologies Section 3. We then show that *Makalu* exhibits both good search performance and scales well for searches in large networks in Section 4. Next we provide some experimental results in Section 5 to validate *Makalu*. In Section 6 we discuss related work and provide concluding remarks in Section 7.

2. Design of the *Makalu* Overlay Algorithm

In this section, we describe the *Makalu* algorithm. We first describe the intuition behind the design of the algorithm and formally describe *Makalu*'s peer rating function.

2.1. *Makalu* Peer Rating Function

The central component of *Makalu* is the peer rating function. Each node makes decisions about accepting connections and pruning existing neighbors based on the result of the peer rating function. The peer rating function computes the relative connectivity and proximity for each neighbor. The connectivity is determined by counting the number of nodes that are reachable through the given neighbor and only that neighbor. The rating function then divides the total number of nodes reachable through all neighbors by this value to determine a relative connectivity value. A higher value indicates a node that offers higher connectivity to the rest of the network. Conversely, if most of the nodes reachable through a given neighbor are reachable through other paths, then that neighbor isn't contributing significantly to the connectivity and its score would be lower. The proximity of each neighbor is computed in a similar manner by dividing the maximum latency to a neighbor by the latency of each neighbor. A lower scores reflect neighbors that are far away and thus have high latency costs. The proximity and connectivity scores for each neighbor are combined and used to rank each neighbor.

Before formally describing the details of how ratings are computed, we first define some terminology. V and E correspond to the set of nodes and edges in a graph respectively. $d_{u,v}$ is the cost (latency) of the edge $(u, v) \in E$ with $u \in V$ and $v \in V$. The neighborhood Γ_u of a node u, specifies the set of nodes directly connected to the given node. The node boundary of a set of nodes, ∂S, represents the collective neighborhoods of all nodes in the set. More specifically, it is the union of all the neighborhoods of the nodes in the set minus the actual nodes in the set. The unique reachable set, $R_{u,v}$, specifies the set of nodes reachable from a node, u, through another node, v, such that those nodes are only reachable from u through v and not through any other of u's neighbors.

Node u computes the rating for a neighbor v using connectivity information from v. It considers the set of nodes that form v's neighborhood, Γ_v, and subtracts from that set any nodes reachable from another of u's neighbors. This forms the unique reachable set from u through v, $R_{u,v}$. The algorithm then computes the ratio of the size of $R_{u,v}$ to the size of $\partial\Gamma_u$, all the nodes reachable through all of u's neighbors (the node boundary of the neighborhood of u). The ratio of the maximum latency to its neighbors, d_{max}, to the latency to v, $d_{u,v}$ is then computed. The algorithm then combines these two ratios and computes a rating for v using the utility function: $F_{u,v} = \alpha\frac{|R_{u,v}|}{|\partial\Gamma_u|} + \beta\frac{d_{max}}{d_{u,v}}$ where d_{max} is the maximum latency to a neighbor of u and α and β are weighting factors for connectivity and proximity respectively. If $\alpha = 1$ and $\beta = 0$, the algorithm is biased toward creating an overlay that is well connected but possibly with poor communication costs. If instead $\alpha = 0$ and $\beta = 1$, the algorithm would create an overlay that has low communication costs at the expense of connectivity. Currently, we give equal weight to both connectivity and proximity and set $\alpha = \beta = 1$ to generate an overlay that treats both connectivity and proximity with equal weight. The resulting ranks are then sorted and the lowest ranking neighbor is disconnected. At each stage the algorithm makes a decision to maximize the utility of its neighbors. Each node seeks to maximize its connection to the P2P network while working within its own capacity constraints. In particular, each node can have different degrees as dictated by its connectivity on the physical network. Having nodes of different degrees doesn't affect the algorithm as each node selects its required number of peers such that the selected peers maximize the utility function F. Additionally, when the degree of a node changes in response to a change in the available bandwidth, the node initiates a pruning mechanism that evaluates its current neighbors using the utility function F and pruning its neighbors with the lowest utility cost until the requisite number of neighbors is reached.

In the following sections we describe the requirements of an overlay to support improved search performance and present an algorithm that uses the peer rating function to create an overlay that meets these requirements.

2.2. *Makalu* Connection Management

Any node joining the overlay needs the address of at least one seed peer. It uses the seed as the starting point of a random walk in order to gather a list of candidate peers. Once a suitable set of candidates is obtained, the node may attempt to establish connections to those nodes in its candidate set until it has obtained sufficient neighbors.

Once a node is connected, it switches to a management phase. In this phase, the node accepts incoming connections as part of its participation in the network. If the number of connections exceeds the maximum number of connections for the node, it proceeds to prune the lower quality connections (neighbors). The ranking of neighbors is done by computing a score for each neighbor using the *Makalu* peer rating function. The basic algorithm for managing connections is described below:

```
Function:Manage()
repeat
    accept connections
    while neighbors > max_connections do
        compute rating for each neighbor
        remove neighbor with lowest rating
    end
until disconnected ;
```

Formally, the *Makalu* algorithm selects neighbors that maximize the node expansion from the local node's neighborhood while minimizing the cost to its neighbors. The algorithm computes a rating for each neighbor that factors in that neighbor's proximity and connectivity to the rest of the network. Preference is given to neighbors that are near (low latency) and neighbors that provide good connectivity to the rest of the overlay. When a node needs to determine whether to make a connection to a new peer, it provisionally considers the candidate peer as its neighbor and computes a rating for all of its neighbors including the candidate peer. The node then keeps the connections with the best rating. The number of neighbors for each node can be controlled by the node itself and reflects the number of connections that the node's current available bandwidth can support.

3. Analysis of *Makalu* Topology

In the previous section we described the *Makalu* algorithm. In this section we show that *Makalu* generated overlays with good connectivity properties. We also show that *Makalu*'s overlays exhibited better fault tolerance and performance than the current Gnutella overlays.

3.1. Analysis Setup

For our experiments, we developed our own network simulator. To validate the topology creation and maintenance mechanisms, we simulated the systems using various physical network models. First, we created a synthetic network model where nodes are assigned coordinates on a plane. The network latency for this model is the Euclidean distance between the nodes. In addition to the Euclidean network model, we used a GT-ITM Transit-Stub [32] network model as well as an artificial network model based on an expanded version of the the all-pairs ping times between PlanetLab node collected by Stribling [28].

We generated *Makalu* topologies using the algorithm described in the previous section. To simulate variable node capacities, we randomly assigned node degrees to each node. Even for large simulations where the number of nodes was greater than 100K, a mean node degree of 10 to 12 was sufficient to obtain good performance. For comparison, we created a randomized power law topology (Gnutella v0.4) using the parameters described in [27, 26] as well as parameters that we extracted from our more current crawls of the Gnutella network and from [29, 24] for the modern Gnutella v0.6 two-tier ultra-peer topologies. k-regular random graphs are expected to be good expanders, but are difficult to maintain in dynamic P2P environments. Therefore, we use k-regular random graphs only as a theoretical optimal graph for comparison purposes. The k-regular random graphs were generated using the algorithm described by Kim et. al. [17].

In the following sections, we compare the path costs and structural properties of *Makalu* overlays to the properties of expander graphs and show the relationship between the structural properties of an overlay and the overlay's tolerance to node failures.

3.2. Graph Diameter and Characteristic Paths

Our aim was to create topologies that were compact with low average communication costs. We used graph diameter and characteristic path length and cost to evaluate our overlay. For this experiment, we computed All-Pairs Shortest Paths between each node in the overlay networks and kept track of cost both in terms of hops and physical network latency. This step is computationally intensive and does not scale well for analyzing networks greater than a few thousand peers. For this reason, we limited the network size to 10,000. The *Makalu* topology had an average shortest path cost of 1205.905 compared to 1629.639 of a k-regular random graph. The Gnutella v0.4 and v0.6 topologies had average path costs of 2915.106 and 1370.809 respectively. Similarly, when we examined the graph diameter, *Makalu* topologies were compact with an average diameter of 5. This was slightly lower than the diameter of a modern two-tier Gnutella topology and a k-regular random graph, both of which had an average diameter of 6. The average diameter of the classic Gnutella power law topology was 16. *Makalu* generated topologies with both low diameter and low average communication costs. In the next section we examine the structural properties of *Makalu*.

3.3. Structural and Connectivity Properties

We next examined *Makalu*'s connectivity properties. Good connectivity is implied by high expansion. However, determining the expansion of a graph is co-NP-complete [8]. Therefore, we turned to spectral graph theory [8, 16, 11] to get approximations for the expansion of a graph. We used the eigenvalue spectrum of the Laplacian matrix representation of a graph [8] to determine the graph's algebraic connectivity. The algebraic connectivity describes *compactness* of a graph; higher values of algebraic connectivity indicate a more connected and compact graph. Given the eigenvalues of the Laplacian of a graph in increasing order, the second smallest eigenvalue, λ_1, is the algebraic connectivity [10]. The algebraic connectivity gives us the following bounds on the vertex expansion of a graph: $\lambda_1(G) \leq v(G) \leq d_{min}(G)$ where $v(G)$ is the vertex connectivity of the graph and $d_{min}(G)$ is the minimal degree of any vertex in G. We can use this to show that a graph with high algebraic connectivity also has high expansion. For this experiment, we computed the Laplacian eigenvalue spectrum of the overlays generated by each algorithm. The algebraic connectivity for a k-regular random graph was 2.7315. The *Makalu* overlay had an algebraic connectivity of 2.7189, close to the ideal of a k-regular random graph. The Gnutella v0.4 topology had a very low algebraic connectivity (0.035); this is a known consequence of its power law structure. The Gnutella v0.6 topology had better connectivity than the v0.4 topology (0.936), but it is significantly lower than *Makalu*'s connectivity.

3.4. Fault Tolerance of *Makalu*

Next, we explored the effect of node failures on the connectivity and properties of the topologies created by the different algorithms. In particular, we are concerned with determining the resiliency of the topologies to the failure of the most highly connected nodes as these nodes represent important nodes in the network. In the interest of space, we present only the results from failing the most highly connected nodes. For this analysis we generated the various topologies and induced a non-recoverable and instantaneous failure of the most highly connected nodes as well as for random nodes. This simulates a worst-case failure model where all failed nodes disappear instantly. The analysis is performed on a snapshot of the overlay immediately after the failure occurs so that the remaining nodes are not given the opportunity to recover. We need to compare the connectivity of graphs of different sizes to evaluate how the loss of nodes affect their connectivity. The standard Laplacian eigenvalue spectrum is not well-suited for this as it can only compare graphs of the same size. Instead, we employ the normalized Laplacian matrix [8] of the graph.

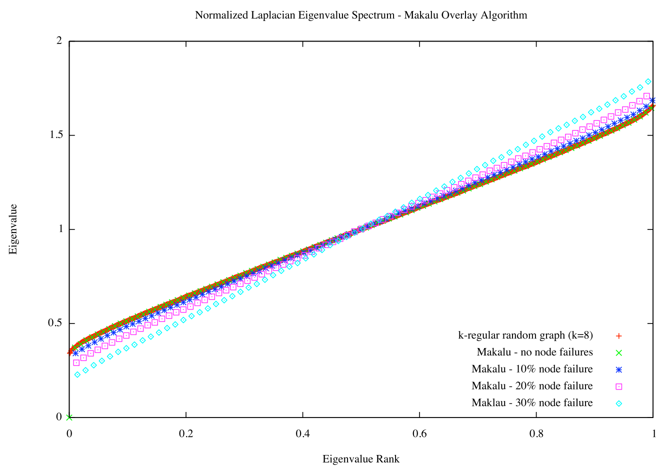

Figure 1. Normalized Laplacian eigenvalue spectrum when the most highly connected nodes fail and are removed from the *Makalu* topology.

The normalized Laplacian matrix constrains the resulting eigenvalues to the interval [0,2]. Each data point is defined to be (x_i, y_i) where x_i is the normalized rank of the ith eigenvalue given by $x_i = \frac{r_i}{n-1}$ and y_i is the ith eigenvalue. The eigenvalues are arranged in increasing order, r_i denotes the rank of the ith eigenvalue. This results in graphs with x values in the interval [0:1] and y values in the interval [0:2]. The multiplicity of eigenvalue 0 describes the number of connected components in the graph [8]. The multiplicity of eigenvalue 1 is related to the number of "edge" nodes that are weakly connected to the network [31]. A higher multiplicity of eigenvalue 1 implies more weakly connected "edge" nodes.

For this experiment, we plotted the normalized Laplacian spectrum of the overlay created by each algorithm. As nodes fail, we preferred the plots to remain similar to the normalized Laplacian spectrum of a k-regular random graph. As shown in Figure 1, the multiplicity of eigenvalue

0 remained at one for the *Makalu* topology. This means that although node failures exist, the overlay remained fully connected. Additionally, we observe that the structure of the *Makalu* topology did not vary significantly with added failures. The remaining nodes did not become weakly connected. We can see this in Figure 1 by noticing that the multiplicity of eigenvalue 1 remained low. Even with 30% node failures, the *Makalu* algorithm maintained a spectrum that was similar to the ideal spectrum of a k-regular random graph. This is important because the structural properties, as we will see in the next section, have implications on the ability of the overlay to support efficient search mechanisms.

4. Search Performance Analysis

In the previous sections we described *Makalu* and showed that it achieved the desirable properties of good connectivity needed to support efficient searches. Under the failure of the most highly connected nodes, a *Makalu* overlay's connectivity did not vary significantly thus allowing the remaining nodes to stay well connected. Additionally, *Makalu* created overlays with good connectivity using only information local to each node. In this section, we show how *Makalu*'s good connectivity yielded improved search performance. We first describe our experimental setup and then present our analysis for flooding and identifier searches.

4.1. System Setup

For our experiments we created a simulator to study the performance of the different search mechanisms on different topologies. We performed our experiments with networks ranging in size from 100 nodes to 100,000 nodes. In this paper, replication ratio represents the percentage of nodes that contain a replica for a given object. Additionally, the nodes that contain a replica for a given object were chosen uniformly at random. We compared our results to search in a classic Gnutella power law topology and to a modern Gnutella two-tier ultra-peer topology. In the case where multiple replicas of an object existed in the system, the query was considered to be successfully resolved if at least one of the replicas was located. We performed 100 separate runs with each run issuing 1,000 queries. Results reported represent the mean of these runs.

4.2. Flooding Performance Analysis

In this section, we analyzed the behavior of controlled flooding on a *Makalu* overlay. For our experiments, we placed data objects uniformly at random on nodes according to the specified replication ratio for each object. Because most objects in file-sharing systems were not highly replicated, we limited our analysis to replication rates of less than 1%. We performed a flooding search for each unique object in the system from random nodes. We used query ID caching for duplicate query suppression. We recorded the number of queries that were successfully resolved as well as the number of messages sent for each query and the number of unique nodes that were visited by the flood. We also tracked the average messages received at each node and the number of replicas located by the query. For flooding on the Gnutella v0.6 (two-tier ultra-peer) topology, we used a modified flooding algorithm that simulates the behavior of current Gnutella query routing. For low replication ratios, a small percentage of queries required a flood to reach almost all nodes in the network. In these cases, the flood needed to propagate beyond reasonable TTL limits and thus resulted in higher mean messages per query. For a network size of 100,000 nodes, we used a TTL for floods that reflects realistic TTL limits observed in Gnutella traces and allow for floods to resolve most ($> 95\%$) of the queries.

We present a summary of these results in Table 1. The comparatively low node degree and high expansion of *Makalu* topologies resulted in fewer messages per query for similar TTL values when compared to Gnutella topologies. Compared to the Gnutella v0.4 power law topology, *Makalu* reduced the TTL required by 50% while reducing the number of messages by a factor of four. The reduction in the number of messages is greater when comparing *Makalu* to the modern Gnutella v0.6 topology under low replication ratios (0.1% and 0.05%). In these cases, the low node degree and high expansion of *Makalu* allowed for most queries to be resolved with fewer than 7,000 messages. The high degree of ultrapeers in the Gnutella v0.6 topology required more than 7 times as many messages. The Gnutella v0.6 topology did reduce the number of messages routed to low-capacity (leaf) peers compared to the earlier v0.4 topology. However, our simulations verify the results presented in [1], which showed that ultra-peers in the modern Gnutella topology are responsible for the majority of the query routing bandwidth of the system. In particular, the high node degree of ultra-peers resulted in many outgoing messages sent per incoming query. This is reflected in the high number of messages required to satisfy queries for the Gnutella v0.6 topology as shown in Table 1. One of the benefits of *Makalu* is that the outgoing message rate per query remained low without affecting the ability to resolve most queries.

4.3. *Makalu* Flooding Efficiency

In the previous section, we showed that *Makalu* flooding successfully resolved queries within a few hops even for low replication ratios. We then focused on the efficiency of *Makalu* flooding. Specifically, we were interested in the

	Gnutella v0.4 (power law)		Gnutella v0.6 (two-tier)		*Makalu*	
Replication Rate	Messages/Query	Min TTL	Messages/Query	Min TTL	Messages/Query	TTL
0.05%	30,557.96	7	51,184.12	4	6,783.32	4
0.1%	24,155.84	7	51,127.22	4	6,668.36	4
0.5%	11,959.16	6	6,444.22	3	769.84	3
1%	11,942.28	6	6,426.56	3	758.48	3

Table 1. Message per query and minimum TTL required to resolve queries on each topology. Network size: 100,000 nodes

number of duplicate messages. *Makalu* had good connectivity between peers (high expansion) while maintaining a low average node degree. These properties allowed flooding queries to be propagated to many nodes with few duplicate messages. With a TTL of 4, a flood on a *Makalu* topology generated approximately 6,500 messages on a 100,000 node network. Of these, only 2.7% were duplicates where the query reached the same node more than once. With a TTL of 4, flooding in *Makalu* resolved all queries for replication ratios above 0.05%. When the replication ratio was set at 0.05% and a TTL of 4 was used, 95% of the queries were satisfied. For relatively high replication ratios ($\geq 0.5\%$), a TTL of 3 resolved all queries with less than 800 messages on a 100,000 node network. Flooding on *Makalu* was efficient, even for low replication ratios (e.g., 0.05%), as most queries were resolved with relatively few messages ($< 10,000$). In addition, the high expansion of *Makalu* reduced the number of duplicate messages sent because each hop in the flood reached unique nodes.

4.4. Flooding Performance Under Low Replication Scenarios

A high expansion implies that from any node, a large percentage of the nodes in the network are reachable within only a few hops. However, the property that allows for such large coverage also implies that a flood that travels deep into the network will generate many duplicate messages. From any node in a graph with high, we expect a large number of disjoint paths leaving the node [12, 13]. As each of these paths reach further out, there are fewer unvisited nodes that each path can reach. This leads to multiple paths converging to the same node. We call the point at which multiple paths begin to reach the same nodes the *Convergence Boundary*. The *Convergence Boundary* occurs when roughly half the nodes have been visited; it coincides with approximately half the diameter of a graph. Flooding in expander graphs have two phases: an expanding phase and a converging phase. While the flood expands toward the *Convergence Boundary*, the flood is in the expanding phase and when it crosses the *Convergence Boundary*, it begins the converging phase. Because paths are disjoint in the expanding phase, flooding generates few duplicate messages in the expanding phase. However, once the flood crosses the *Convergence Boundary*, there will be a significant increase in the number of duplicate messages. Epidemic algorithms [30, 9, 19] might be deployed beyond the *Convergence Boundary* to reduce the number of such duplicates. For realistic replication ratios, *Makalu* flooding is efficient. In circumstances where the replication ratio is very low and every node must be reached, a DHT-based flooding mechanism such as Structella [5] may give better performance. Nevertheless, even for a replication ratio such as 0.01% (or 10 nodes out of 100,000), flooding on *Makalu* resolved 56% of queries within 4 hops and an approximately 6,500 messages.

4.5. Scalability of Flooding as the Size of the Network Increased

In the previous section, we showed that we can exploit the properties of a *Makalu* topology to perform an efficient flooding search. In this section we show the performance of *Makalu* flooding on networks of various sizes. We were interested in analyzing the interplay between the minimum TTL requires to satisfy queries and the number of messages generated per query as the size of the network increased. In this experiment, we fixed the replication ratio of objects and varied the size of the network. Although we only present results for a replication ratio of 1%, our results for various replication ratios from 0.05% to 0.5% showed similar results. In Figure 2 we plotted the average number of messages sent per query as the size of the network increased using a fixed replication ratio of 1% and a fixed TTL of 4. Note that this is a log-log plot. We see evidence that even though the number of messages increased as the network grew, floods on larger networks sent proportionally fewer messages with respect to the size of the network. The number of messages sent grew slower than linearly, which means that flooding on a *Makalu* topology scaled well as the size of the network increased.

In Figure 3, we show how the relationship between suc-

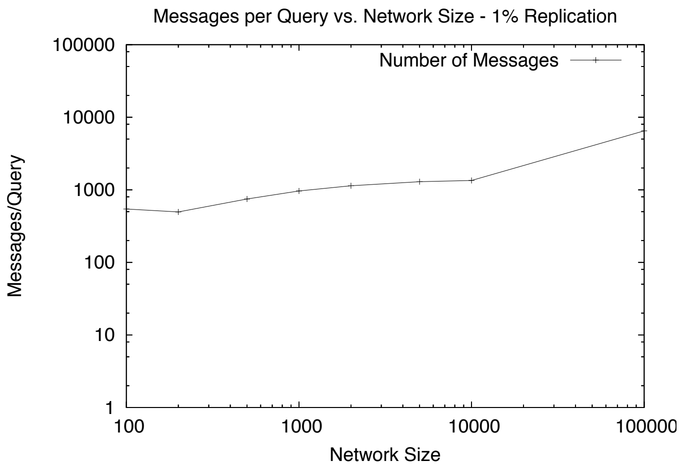

Figure 2. Messages per query vs. network size for various *Makalu* network sizes using a 1% replication ratio ($\log$-$\log$ plot). The messages per query use a fixed TTL of 4 which yields a 100% success rate in resolving queries.

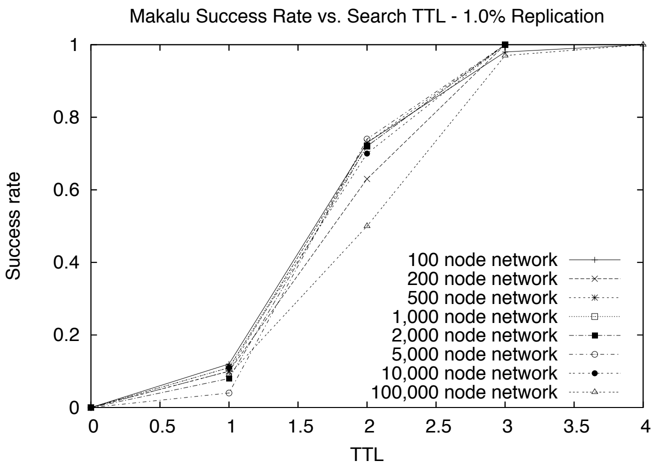

Figure 3. Success rate vs. TTL of flooding search for various *Makalu* network sizes using a 1% replication ratio.

cess rate and TTL changes as the network size changes. We see that the success rates were similar across all network sizes. This behavior was expected of the *Makalu* topology. Because the capacity of each node remained fixed as the network grew, smaller networks had limited opportunity to maximize their connectivity. So, even though larger graphs had more nodes, their "spread" within the first few hops was not restricted. Floods on larger graphs reached proportionally more nodes at each hop. This means that although larger graphs sent more messages, the flood did not need to travel as far because it was likely to locate a replica in fewer hops.

4.6. Indexed Identifier Search

In this section, we analyze the performance of the system in resolving searches where the identifier of the desired object is known in advance. We show that the expander-graph properties of our overlay led to favorable conditions for success of this approach. We examined indexed identifier searches using attenuated Bloom filters [25]. A Bloom filter is a compact representation of a large set of objects that allows one to easily test whether a given object is a member of that set [4]. An attenuated Bloom filter is a hierarchy of Bloom filters, each of which contains aggregate information about some set of nodes. Specifically, the Bloom filter at level i represents the aggregate content store on nodes that are i hops away. Because deeper levels of the attenuated Bloom filter contain aggregate information for an increasingly larger set of nodes, the corresponding Bloom filter will be more populated thus increasing the false positive rate. For this reason, the results from Bloom filters near the top of the hierarchy are given more weight when computing the score because their false positive rate is expected to be low. The high expansion properties of the *Makalu* overlay allowed the attenuated Bloom filter to function efficiently since each node had access to information about a large number of nodes. The attenuated Bloom filter captured information about a large portion of the network while limiting the number hops in the network needed to obtain this information. When two peers established a connection, they exchanged routing tables and their corresponding attenuated Bloom filters. The routing tables and attenuated Bloom filters capture information about each peers neighborhood. In this way, peers need only communicate with their direct neighbors to discover information about their neighborhood. Searches using attenuated Bloom filters were resolved quickly because at each hop in the search, the potential function guiding the search was able to make high quality decisions.

For this experiment, we generated a 100,000 node topology and populated the nodes with objects where each object was replicated uniformly at random on a given percentage of the nodes. We then ran the search algorithm using an attenuated Bloom filter [25] with a depth of three on our simulator and recorded the number of messages sent, number of nodes that were visited, and the number of queries that were successfully resolved. Figure 4 plots the percentage of successful queries versus the TTL used for each query.

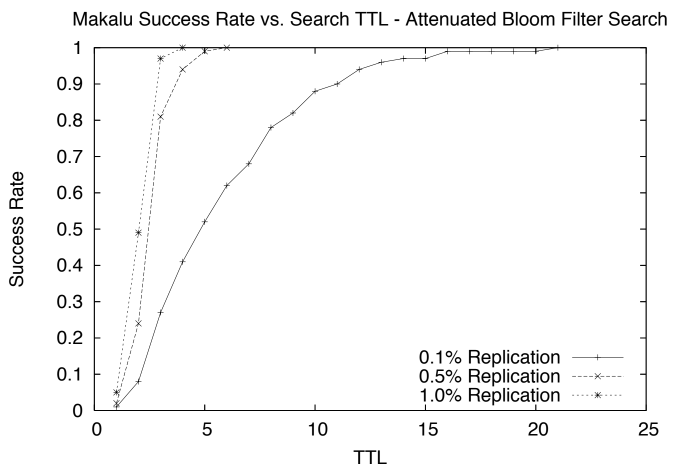

Figure 4. Success rate vs. TTL of attenuated Bloom filter search for various replication ratios using *Makalu* on a 100,000 node network.

For relatively high replication rates (0.5% and 1%), most queries (>95%) were resolved in less than five hops and all queries were resolved within 8 hops. For a lower replication rate of 0.1%, most queries (> 75%) were resolved within ten hops and over 95% of queries were resolved within 15 hops. Identifier searches in *Makalu* were efficient because the attenuated Bloom filter at each node captured the content contained in a large number of nodes; this is a consequence of *Makalu*'s expander graph properties.

4.7. Summary of Results

In this section we showed that a *Makalu* overlay effectively supported efficient searches. We showed that *Makalu* resolved flooding searches with fewer messages per query than Gnutella. Further, we showed that *Makalu* scaled well as the number of nodes in the overlay increased. We showed that the number of messages per query increased sub-linearly with the size of the network. Increasing the network size by two orders of magnitude only increased the number of messages per query by about 2.6 times. In addition to flooding searches, we showed that *Makalu* supported efficient identifier searches. We used attenuated Bloom filters for probabilistic routing of identifier queries and showed that even on networks as large as 100K nodes, most queries were resolved in fewer than 10 hops. In the next section, we validate *Makalu* against data collected from the state of Gnutella in 2003 and 2006.

5. Experimental Validation

In addition to simulation results, we were interested in evaluating our design choices against a current active and large-scale P2P system. The analysis in [1] provides data on query rates and bandwidth utilization of Gnutella traffic in 2003 and 2006. In 2003, a peer received over 400K query messages in a 2 hour interval, or approximately 60 queries per second. In 2006, this number was drastically reduced to 23K queries in a 2 hour interval, or about 3 queries per second. However, in 2006, queries were propagated by ultra-peers to a mean of 38 peers, compared to 4 peers in 2003. This corresponds to an outgoing query bandwidth of 103 kbps in 2006, compared to over 130 kbps in 2003 resulting in little net bandwidth savings compared to the large reduction in incoming query traffic.

	Gnutella	*Makalu*
Outgoing Msgs per query	38.439	8.5
Outgoing Msgs per second	124.16	27.45
Outgoing Bandwidth	103.4 kbps	23.04 kbps
Query Success Rate	6.9%	36%

Table 2. Traffic comparison between *Makalu* and Gnutella Search Traffic

Applying these findings to flooding search on a *Makalu* topology resulted in a significant improvement of both search performance and bandwidth utilization. Using the current Gnutella incoming query traffic metrics, we calculated the bandwidth used for a flooding search on a *Makalu* topology with 100,000 nodes and mean node degree of 9.5. We evaluated *Makalu* searches on our simulator assuming a worst case scenario where each object existed on only 1 node in the 100,000 node network. The measured success rates captured the worst case performance of search. On average, a search had to cover a large portion of the network. The results are shown in Table 2. With a mean incoming query traffic rate of 3.23 queries per second and a mean query size of 106 bytes, a search on a *Makalu* topology generated 8.5 outgoing messages per query and 0.9 kB per query or 23.04 kbps. Using a TTL of 5, simulation results showed that in a 100,000 node network with mean node degree of 9.5, flooding searches in *Makalu* resolved 36% of the queries. In comparison, the current Gnutella protocol can generate up to 103.4 kbps of outgoing query traffic and the success rate experienced by the traffic capturing client was only 6.9%. Currently, a Gnutella ultra-peer can have many neighbors; our traffic capturing client had up to 64 neighbors with 35 to 40 ultra-peer neighbors active at any given point in time. Yet, such a large number of neighbors only yields a success rate visible to the client of less than 10%. A similar search on *Makalu* resolved 36% of the queries with 75% less neighbors per node and, in turn, 75% less outgoing bandwidth consumption.

6. Related Work

Previous research has investigated various search techniques in unstructured P2P networks. For example, random walks have been proposed as an alternative to flooding in unstructured P2P networks [20]. Random walks can effectively reduce the number of messages sent per query at the expense of increased response time for the query. Search performance using random walks depends on an overlay having connectivity properties that allow for the walk to traverse the overlay efficiently. Other work [2] relied on searches being routed to the highly connected nodes since these nodes have access to the largest number of neighbors. However, this approach placed a great burden on these highly connected nodes as they must route the majority of queries. Fundamentally, searches required access to information at each node in order to effectively be resolved. An overlay with low connectivity at most peers as in power law topologies lacks the ability to provide searches with the necessary information to resolve the queries efficiently. Gia [7] attempted to improve the scalability of power law systems by choosing high capacity nodes for immediate peers and replaced the flooding search with a random-walk search. However, Gnutella's topology is no longer a power law topology thus limiting Gia's effectiveness in Gnutella-like systems. Additionally, power law topologies are susceptible to targeted attacks. *Makalu*'s design was geared to providing fault-tolerance and facilitating current and future searches in unstructured P2P with little modification to the search mechanisms.

Our work showed that *Makalu* resolved most queries given a reasonable TTL. However, we do not mandate a specific mechanism for selecting the TTL. The TTL may be set as a parameter of the system as in the current Gnutella. Alternatively, a dynamic TTL selection mechanism can be used to adapt the TTL for queries at run-time. Chang and Liu in [6] described a dynamic programming mechanism that selected an appropriate TTL when the probability distribution of the object locations was known in advance. When the distribution was not known in advance, they used a randomized mechanism to select an appropriate TTL. This approach can be integrated into a *Makalu* search that relies on TTL to control the spread of queries.

Early work in analyzing peer-to-peer systems identified several key features of P2P file-sharing systems. First, Saroiu et. al. [27] and Ripeanu et. al. [26] studied the Gnutella file-sharing network to determine the structure of the Gnutella topology and the characteristics of the network connectivity and file-sharing behavior of users. They found that a large percentage of nodes had slow and low bandwidth network connectivity. Additionally, they reported that the overlay topologies have power law degree distributions. More recently, Stutzbach et. al. [29] show that the modern Gnutella two-tier ultra-peer architecture does not follow a true power law distribution since ultrapeers try to maintain a fixed number of connections. In [23] Bustamante et al characterized the behavior of Gnutella [14] and Overnet/eDonkey [22]. The authors showed that Gnutella's queuing time was significantly slower than Overnet's due, in part, to Overnet's use of a distributed hash table (DHT) [21] to speed up keyword lookups. Our prior study of the long-term analysis of Gnutella traffic [1] revealed that more than 90% of the bandwidth used to process queries at an ultra-peer is wasted. Our goal in this work was to develop a P2P system that can improve search performance by creating an overlay that has good connectivity similar to expander graphs.

Law et. al. [18] explored the use of expander graphs as overlays for P2P networks. Their approach involves the use of $2d$-regular multigraphs where the edges of the graph are composed of d Hamilton cycles. They prove that their algorithm generates an overlay that is an expander graph with high probability. Although regular random graphs are theoretically good expanders, creating a P2P system using k-regular random graph on real networks poses several problems. First, building and maintaining regular random graphs in a dynamic and distributed network is expensive and requires a great deal of coordination and communication. Routing on such an overlay depends on the overlay being stable. In the presence of node churn, maintaining a stable overlay that is a k-regular random graph becomes increasingly complex. Second, the uniform node degree restriction of k-regular graphs does not map well to real networks. There can be large variability in the network connectivity of the nodes in the network. Forcing a fixed degree causes some nodes to be over-committed while communication capacity of other nodes are under-utilized. Instead, our algorithm relies only on local information for each node and tries to maximize the expansion from each node's neighborhood while minimizing the latency to its neighbors. We show that the resulting overlays exhibit similar connectivity properties to expander graphs without requiring global coordination or a fixed number of connections at each node.

7. Conclusion

In this paper, we showed that a distributed algorithm can create overlays that approximate the structure of expander graphs using only local information while allowing nodes to have varying degrees. We showed that this algorithm was able to withstand the failure of over 30% of the nodes in the system while still maintaining good communication costs and search performance. For low replication ratios, *Makalu* resolved most queries within 4 hops and 6,500 messages in a 100,000 node network with less than 5% of the messages sent being duplicate messages. We also showed that *Makalu* scaled well to over 100,000 nodes with respect to flooding search performance. Additionally, *Makalu* overlays can be exploited for their high expansion to support efficient indexed identifier searches using mechanisms such as attenuated Bloom filters to route the queries. In this approach, most searches were resolved in less than 10 messages (hops) for replication ratios as low as 0.01%. We also showed experimentally that *Makalu* achieved higher query success rates than Gnutella searches while reducing the bandwidth consumed at each node.

References

[1] W. Acosta and S. Chandra. Trace driven analysis of the long term evolution of gnutella peer-to-peer traffic. In *Proceedings of the Passive and Active Measurment Conference (PAM'07)*, 2007.

[2] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman. Search in power-law networks. *Physical Review E*, 64, 2001.

[3] N. Alon. Eigenvalues and expanders. *Combinatorica*, (6):83–96, 1986.

[4] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[5] M. Castro, M. Costa, and A. Rowstron. Should we build gnutella on a structured overlay? In *Proceedings from the 2nd Workshop on Hot Topics in Networks (HotNets-II)*, 2003.

[6] N. Chang and M. Liu. Revisiting the ttl-based controlled flooding search: optimality and randomization. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 85–99. ACM Press, 2004.

[7] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making gnutella-like p2p systems scalable. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 407–418. ACM Press, 2003.

[8] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[9] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenkcr, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *PODC '87: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12, New York, NY, USA, 1987. ACM Press.

[10] M. Fiedler. Algebraic connectivity of graphs. *Czech. Math J.*, 23:298–305, 1973.

[11] A. Frangiono and S. S. Capizzano. Spectral analysis of (sequences of) graph matrices. *SIAM Journal of Matrix Analysis and Applications*, 23(2):339–348, 2001.

[12] A. M. Frieze. Edge-disjoint paths in expander graphs. *SIAM Journal on Computing*, 30(6):1790–1801, 2001.

[13] A. M. Frieze and L. Zhao. Optimal construction of edge-disjoint paths in random regular graphs. *Combinatorics, Probability and Computing*, 9(3):241–263, May 2000.

[14] Gnutella protocol v0.4. http://dss.clip2.com/GnutellaProtocol04.pdf.

[15] Gnutella protocol v0.6. http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html.

[16] N. Kahale. Eigenvalues and expansion of regular graphs. *Journal of the ACM*, 42(5):1091–1106, 1995.

[17] J. H. Kim and V. H. Vu. Generating random regular graphs. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 213–222. ACM Press, 2003.

[18] C. Law and K.-Y. Siu. Distributed construction of random expander networks. In *IEEE INFOCOM*, 2003.

[19] C. Lindemann and O. P. Waldhorst. Exploiting epidemic data dissemination for consistent lookup operations in mobile applications. *SIGMOBILE Mob. Comput. Commun. Rev.*, 8(3):44–56, 2004.

[20] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th international conference on Supercomputing*, pages 84–95. ACM Press, 2002.

[21] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *IPTPS '02: 1st International Workshop on Peer-to-Peer Systems*, 2002.

[22] Overnet. http://www.overnet.org/.

[23] Y. Qiao and F. E. Bustamante. Structured and unstructured overlays under the microscope - a measurement-based view of two p2p systems that people use. In *Proceedings of the USENIX Annual Technical Conference*, 2006.

[24] A. H. Rasti, D. Stutzbach, and R. Rejaie. On the long-term evolution of the two-tier gnutella overlay. In *IEEE Golbal Internet*, 2006.

[25] S. C. Rhea and J. Kubiatowicz. Probilistic location and routing. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, June 2002.

[26] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6(1), 2002.

[27] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, San Jose, CA, USA, January 2002.

[28] J. Stribling. Planetlab all pair ping data. http://www.pdos.lcs.mit.edu/∼strib/pl_app/.

[29] D. Stutzbach, R. Rejaie, and S. Sen. Characterizing unstructured overlay topologies in modern p2p file-sharing systems. *IEEE/ACM Transactions on Networking*, 2007.

[30] W. Vogels, R. van Renesse, and K. Birman. The power of epidemics: Robust communication for large-scale distributed systems, 2003.

[31] D. Vukadinovic, P. Huang, and T. Erlebach. On the spectrum and structure of internet topology graphs. In *In proceedings of Innovative Internet Community Systems (I2CS)*, Khlungsborn, Germany, June 2002.

[32] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to model an internetwork. In *IEEE Infocom*, volume 2, pages 594–602, San Francisco, CA, March 1996. IEEE.